ภาษา IQ0

"ที่ผมสร้างภาษาขึ้นมาใหม่ไม่ใช่เพราะมีความจำเป็นอันใด หรือประสงค์จะออกแบบภาษาใหม่ หากเป็นเพราะผมต้องการทราบว่า การเขียนโปรแกรมตัวแปรภาษาเขาทำกันได้อย่างไร"

บุญเลิศ เอี่ยมทัศนา

ภาษานี้อ่านเป็นคำเต็มว่า **"ภาษาไอคิวซีโร"** หากคุณคิดว่าผมกำลังนำภาษาใหม่มาเสนอเช่นเคยก็ถูกต้องครับ แต่ไม่ถูกต้องนัก เพราะภาษานี้จะว่าเก่าก็ไม่ใช่ ใหม่ก็ไม่เชิง เพราะผมนำมาจากภาษาปาสคาลและภาษาซีมารวมกัน สร้างขึ้นเป็นภาษาใหม่ ขีดความสามารถของภาษาก็สมกับชื่อครับ

โปรแกรมตัวแปรภาษา

ที่ผมสร้างภาษาขึ้นมาใหม่ไม่ใช่เพราะมีความจำเป็นอันใด หรือประสงค์จะออกแบบภาษาใหม่ หากเป็นเพราะผมต้องการทราบว่า การเขียนโปรแกรมตัวแปรภาษาเขาทำกันได้อย่างไร หาหนังสือเกี่ยวกับเรื่องนี้มาหลายเล่มแต่ไม่เคยอ่านจบ เพราะอ่านของเขาไม่เข้าใจ จึงคิดว่าน่าจะทดลองเขียนตามแนวความคิดอย่างทื่อๆ (คือโง่ๆ) ของตนเองก่อนนี้ แล้วนำไปเทียบเคียงกับของเขาอาจจะทำให้เข้าใจได้ จึงเกิดเป็นภาษา IQ0 ขึ้นมาครับ

ส่วนที่นำมาเสนอก็เพราะ เห็นว่าน่าจะเป็นประโยชน์แก่คุณผู้อ่านบ้างไม่มากก็น้อยในการทำความเข้าใจเพิ่มเติมกับภาษาปาสคาลและซี เช่นความหมายของคำว่า reserved word และคำว่า standard identifiers คืออะไรกันแน่ รวมทั้งวิธีเขียนโปรแกรมตัวแปรภาษา ซึ่งถึงแม้จะไม่ถูกต้องทีเดียวก็คงพอทำให้มองเห็นแนวทาง

สำหรับคุณที่ไม่ได้อ่าน "ภาษา-เครื่องสำหรับภาษาสูง" ในไมโครฯ 32 กรุณาอ่านด้วยครับ เพราะเรื่องนี้กลายเป็นตอนต่อของเรื่องนั้นไป (ทั้งที่ว่าจะไม่เขียนต่อครับ)

(ซีพียู+ภาษา) สมมติ

ดังกล่าวแล้วว่า ซีพียูของผมเป็นซีพียูสมมติ และก็แตกต่างกับซีพียูทั่วไปด้วยคือ มีรีจิสเตอร์เพียง 2 ตัว ได้แก่ PC (Program Counter) ไว้ชี้คำสั่ง กับ SP (Stack Pointer) ไว้ชี้ชั้นยอดของแสต็ค ไม่มีรีจิสเตอร์ใช้งานทั่วไป หากใช้สแต็คดำเนินการแทน แถมผมได้ตกลงใจแล้วว่าไม่ต้องมีแฟลก ที่แปลกออกไปอีกคือ แยกหน่วยความจำสำหรับเก็บคำสั่งเครื่องกับหน่วยความจำสำหรับเก็บข้อมูลออกจากกันอย่างเด็ดขาด และจากบทความที่กล่าวถึง จะเห็นได้ว่าคุณสามารถนำซีพียูสมมตินี้ไปติดตั้งบนเครื่องใดก็ได้ เช่น แอปเปิลทูหรือไอบีเอ็ม-พีซี

พอมาถึงภาษา IQ0 ก็เป็นภาษาสมมติอีกเช่นกัน เมื่อมองในแง่ของการซ้อนกันคือ ภาษาส่วนมากจะเขียนบนระบบจัดการโดยตรง แต่ IQ0 เขียนอยู่บนภาษาปาสคาลอีกทอดหนึ่ง ซึ่งความจริงก็เป็นวิธีการเขียนโปรแกรมตัวแปรภาษาแบบหนึ่ง หากจะเพิ่มรหัสกลาง (Ncode) ให้เพียงพอแก่การใช้งาน เช่น ในขณะนี้ไม่มีรหัสกลางเกี่ยวกับไฟล์เลย และถ้าจะให้เป็นจริงยิ่งขึ้น โปรแกรมที่ผมเขียนด้วยภาษาปาสคาลในบทความนี้ จะต้องเปลี่ยนเป็นเขียนด้วยรหัสกลางทั้งหมด เมื่อจะนำไปใช้กับเครื่องนั้นๆ และบนระบบจัดการนั้นๆ แต่นี่เราเพียงแต่ทดลองก็คงสมมติกันต่อไปครับ

ภาษากับการออกแบบ

ดังที่กล่าวแล้วว่า ภาษา IQ0 เป็นแบบปาสคาลผสมซี นั่นคือ ผมไม่ได้ออกแบบภาษานี้ใหม่ ตรงนี้ผมต้องกล่าวให้กระจ่างครับคือ ถ้าผมเขียนเรื่องนี้เมื่อประมาณ 20 ปีที่แล้ว คือก่อนที่มีภาษาปาสคาลและซี ผมก็จะได้ชื่อว่าเป็นผู้สร้างภาษา IQ0 แต่เมื่อมาเขียนในวันนี้ไม่ใช่ครับ เพราะผมได้ลอกเลียนหลักไวยากรณ์ (Syntax) ของสองภาษานี้มาตั้งเป็นภาษาสมมติขึ้นเพื่อเป็นตุ๊กตาว่าการเขียนโปรแกรมตัวแปรภาษาเขียนได้อย่างไร

ผู้ที่ออกแบบภาษาเทียบได้กับสถาปนิกออกแบบบ้าน ส่วนผู้ที่นำไปอนุวัติ (implement) ให้เป็นจริง คือ นำไปเขียนเป็นโปรแกรมตัวแปรภาษาเทียบได้กับช่างก่อสร้างหรือวิศวกร ในวงการก่อสร้างทั้งสถาปนิกและวิศวกรดูเหมือนจะได้รับการยกย่องเท่าๆกัน แต่ในวงการคอมพิวเตอร์ ผู้ที่ได้รับการยกย่องคือผู้ที่ออกแบบภาษา และดูเหมือนว่า ผู้ที่ออกแบบภาษาหรือทีมงานของตนจะเป็นผู้เขียนโปรแกรมตัวแปรภาษาตัวแรกของภาษานั้น หากได้รับความนิยมก็จะมีองค์การหรือบริษัทอื่นนำไปอนุวัติ เพื่อใช้กับเครื่องอื่นๆต่อไป

สิ่งที่เหมือนกันคือ สถาปนิกอาจออกแบบบ้านไว้ดี แต่ผลงานขึ้นกับช่างก่อสร้าง ในวงการคอมพิวเตอร์ก็เช่นกัน จึงมีการกล่าวกันว่า ภาษานี้ของบริษัทนั้นดีกว่าของบริษัทโน้น เป็นต้น

สำหรับในเรื่องของภาษา ไม่ว่าจะเป็นภาษาคอมพิวเตอร์หรือภาษามนุษย์ย่อมประกอบไปด้วยประโยค ซึ่งในภาษาคอมพิวเตอร์ประโยคอาจประกอบด้วยสัญลักษณ์ คำ และสตริง (ดูเหมือนจะเป็น ดร.เวียร์ต ผู้ออกแบบภาษาปาสคาลได้กล่าวว่า ไม่ควรใช้คำว่า ภาษา คือ Language กับ คอมพิวเตอร์ เพราะมันเป็นแต่เพียงสัญลักษณ์ที่ป้อนเข้าเครื่องเท่านั้น แต่เมื่อใช้กันมาแล้วก็คงต้องใช้ต่อไป ซึ่งผมก็เห็นด้วย และคงต้องใช้ต่อไปเช่นกันครับ)

เมื่อมองในทัศนะของภาษา ในภาษามนุษย์มีหลักภาษา ในภาษาคอมพิวเตอร์ก็มีหลักภาษาเช่นกัน โดยที่ผู้ออกแบบภาษา จะกำหนดเป็น Syntax กับ Semantic เพื่อให้ผู้ที่นำไปอนุวัติหรือผู้ใช้ภาษานั้นเข้าใจ (ผู้อนุวัติภาษาปาสคาลรายหนึ่ง คือ บริษัทบอร์แลนด์ เรียกว่า เทอร์โบปาสคาล ผู้ใช้คือคุณและผม) ความหมายของคำ 2 คำ นี้ ผมขอประมวลมาจากหนังสือที่กล่าวถึงในรายการหนังสือท้ายเรื่องนี้คือ

ไวยากรณ์ (Syntax)

เป็นการกำหนดรูปแบบของประโยค เช่น ประโยคในภาษาอังกฤษ (รูปแบบหนึ่ง) จะประกอบด้วย ประธาน กริยา และกรรม หากผู้ใดเขียนผิดไปจากนี้จะถือว่าผิด เช่น ประโยคว่า 'The wise man avoids computers' ถือว่าถูก แต่ถ้าเขียนว่า 'The avoids wise man computers' ถือว่าผิด Syntax

คำว่า Syntax ในภาษาไทยใช้คำว่า "ไวยากรณ์" ซึ่งผมก็ไม่ทราบว่าเหมาะสมหรือไม่ เพราะความหมายของมันคือ กฎเกณฑ์หรือข้อกำหนดของภาษาแต่เมื่อใช้กันมาแล้ว ก็คงต้องใช้ต่อไปครับ ผู้ออกแบบภาษาจะต้องวางข้อกำหนดหรือไวยากรณ์ของภาษาในรูปแบบที่เข้าใจได้ง่าย ที่นิยมในระยะหลังนี้ คือ BNF (Backus-Naur Form หรือ Backus Normal Form) กับ Syntax diagram ซึ่ง Syntax diagram เข้าใจง่ายกว่า แต่เป็นภาพเปลืองหน้ากระดาษ ผมจึงจะใช้ BNF

ซีแมนติก (Semantic)

คือนิยามของ ไวยากรณ์ที่ตั้งขึ้น เช่น กำหนดไวยากรณ์ของประโยค IF ไว้ว่า ต้องขึ้นต้นด้วยคำว่า 'IF' ตามด้วยเงื่อนไข คำว่า 'THEN' อนุประโยค ปิดท้ายด้วยคำว่า 'ENDIF' ที่ต้องแสดงนิยามของประโยคนี้ไว้ว่า "เมื่อเงื่อนไขเป็นจริง จึงจะทำอนุประโยค ซึ่งอยู่ระหว่าง 'THEN' กับ 'ENDIF' หากเงื่อนไขเป็นเท็จจะไม่ทำงานในประโยคนี้เลย" ดูไม่จำเป็นนะครับ ทั้งนี้เพราะเราทราบแล้ว แต่ถ้าไม่เคยมีรูปแบบของประโยค IF มาก่อน หรือกับผู้ที่ไม่เคยศึกษาภาษาคอมพิวเตอร์มาก่อน นิยามหรือคำอธิบายนี้จะจำเป็นมาก

ในปัจจุบันยังไม่มีผู้ใดออกรูปแบบของการนิยามให้กะทัดรัดเข้าใจง่าย ดังเช่นการกำหนดไวยากรณ์ คงเขียนเป็นคำอธิบายธรรมดา และเนื่องจากตามที่กล่าวแล้วว่า ภาษา IQ0 เลียนแบบภาษาปาสคาล ผมจึงไม่จำเป็นต้องเขียนนิยามของภาษา IQ0 ครับ

ไวยากรณ์ของ IQ0

ผมได้กล่าวว่า ผมจะแสดงไวยากรณ์ของภาษา IQ0 ด้วยวิธีของ BNF แต่เมื่อทำจริง เห็นว่าการทำความเข้าใจกับแบบฟอร์มของ BNF เป็นเรื่องยาวพอสมควร ผมจึงเสนอในแบบที่ให้เข้าใจง่ายๆ ดังแสดงในรูปที่ 1 ขอแสดงความหมายของสัญลักษณ์ก่อน ซึ่งจะมีความ

หมายดังนี้

→	ประกอบด้วย
< >	ให้เลือกได้
,	และ (ซึ่งจะต้องเรียงลำดับกันด้วย)
;	หรือ
!	อันใดอันหนึ่ง
( )	เลือกภายในวงเล็บเป็นหนึ่งเดียว
' '	เป็นคำบังคับให้ใช้ตามนี้

ในการอ่านไวยากรณ์นี้ ผมขอทำความเข้าใจเสียก่อนว่า คำต่างๆ ในรูปที่ 1 นั้น เป็นคำที่เราคือผมกำหนดขึ้น ไม่ได้หมายความว่า จะต้องเขียนตามนี้ ส่วนความหมายจะพ้องหรือไม่พ้องกับความหมายที่แท้จริงก็ได้ ดังนั้น คำทุกคำจึงต้องนิยามเป็นทอดๆไป

เช่นในบรรทัดแรก จะอ่านได้ความว่า program ประกอบด้วย heading, declaration, block, main และคำว่า ENDPRO ซึ่ง heading, declaration และ block จะมีหรือไม่ก็ได้ ส่วนคำว่า ENDPRO เป็นคำบังคับต้องมีและเขียนตามนี้ (จะเป็นตัวเล็กหรือตัวใหญ่ก็ได้ตามหลักภาษาปาสคาล) ซึ่งเมื่ออ่านบรรทัดนี้แล้ว จะไม่เข้าใจว่า คำต่างๆ เหล่านี้หมายถึงอะไร ก็ต้องอ่านต่อไป เช่น

คำว่า heading จะอ่านได้ความว่า heading ประกอบด้วยคำว่า PROGRAM และ identifier และเครื่องหมาย ; เมื่อไปอ่านคำว่า identifier ต่อไป จะได้ความว่า ประกอบด้วย letter และอาจจะต่อด้วย letter หรือ digit หลายๆ ตัวได้ เมื่อไปอ่านคำว่า letter จะได้ความว่า ประกอบด้วยอักขระ a ถึง z ทั้งอักษรเล็กและใหญ่กับขีดล่างตัวใดตัวหนึ่ง ส่วน digit ก็เช่นกัน ประกอบด้วยอักขระ 0 ถึง 9 ตัวใดตัวหนึ่ง

ผมจะไม่อ่านทั้งหมดขอกล่าวเฉพาะตรงที่น่าสงสัยเช่น คำว่า statement มีเครื่องหมายให้เลือกเช่นเดียวกับคำว่า program แต่ใช้เครื่องหมาย ; คือ "หรือ" เชื่อมต่อ ส่วน program ใช้เครื่องหมาย , คือ "และ" เชื่อมต่อ ความหมายก็คือ ใน program เมื่อเลือกแล้วจะต้องเรียงลำดับตามนี้ ส่วนใน statement จะเรียงลำดับอย่างไรก็ได้ สำหรับใน expression ภายในวงเล็บไม่ได้ใช้เครื่องหมาย "อันใดอันหนึ่ง" เพราะหมายถึงให้รวมทั้งหมดนี้คือ operan และ sign และ operan เป็นหนึ่งเดียว คำที่มีอักษร s ต่อท้าย หมายถึงจะให้ซ้ำกี่ครั้งก็ได้

จะเห็นว่า Syntax form ของผมดูขัดๆชอบกล นำมาแสดงก็เพียงให้พอรู้

```
program  -->  <heading>,<declaration>,<block>,main,'ENDPRO'
heading  -->  'PROGRAM',<identifier>

declaration  -->  <IntDeclare>,<CharDeclare>,<StringDeclare>
IntDeclare  -->  'INT',identifiers,';'
CharDeclare  -->  'CHAR',identifiers,';'
StringDeclare  -->  'STRING',identifier[number]s,';'

block  -->  identifier<(identifiers)>,statements,'ENDFTN'
main  -->  'MAIN',statements,'ENDFTN'

identifiers  -->  identifier,< ',' ,..,identifier>
identifier[number]s  -->  identifier[number],< ',' ,..,identifier[number]>
identifer  -->  letter(<letters>;<digits>)
letters  -->  letter<..letter>
digits  -->  digit<..digit>

statements  -->  statement,<..statement>
statement  -->  <assignment>;<control>;<command>
assignment  -->  identifier,' = ',expression,';'
expression  -->  operan;(operan,sign,operan)
operan  -->  (identifier|number)
control  -->  <if>;<ifelse>;<while>
if  -->  'IF',condition,'THEN',statements,'ENDIF'
ifelse  -->  'IF',condition,'THEN',statements,'ELSE',statements,'ENDIF'
while  -->  'WHILE',condition,'DO',statements,'ENDY'
condition  -->  identifier,compare,operan
commmand  -->  standard;function
standard  -->  readint(number);writeint(number);readchar(char);
               writechar(char);readstr(string);writestr(string);nl
function  -->  block

number  -->  digit<..digit>
char  -->  ASCII(20h|..|FFh)
string  -->  char<..char>

letter  -->  (a|..|z:A|..|Z|_)
digit  -->  (0|..|9)
sign  -->  (+|-|*|/)
compare  -->  (=|>|<|#)
```

รูปที่ 1
ไวยากรณ์ของภาษา IQ0

เพื่อเขียนโปรแกรมตัวแปรภาษาต่อไปเท่านั้นครับ สำหรับตัวภาษา IQ0 ก็แตกต่างกับปาสคาลมาก เริ่มจากกำหนดให้จบโปรแกรมด้วยคำว่า ENDPRO แทนที่จะเป็น END. แบ่งโปรแกรมออกเป็นบล็อกเช่นเดียวกับปาสคาล แต่ตามแบบภาษาซีคือ จะซ้อนบล็อกกันไม่ได้ และกำหนดให้มีบล็อก main ตามแบบภาษาซี แต่เนื่องจากผมต้องการให้โปรแกรมตัวแปรภาษาเป็นแบบ one pass คือ แปลเพียงเที่ยวเดียว จึงกำหนดให้ main ซึ่งจะเป็นโปรแกรมหลักอยู่ท้ายสุด

ในแต่ละบล็อกนี้ ได้กำหนดให้ขึ้นต้นด้วยชื่อบล็อกตามอย่างภาษาซี แทนที่จะขึ้นต้นด้วยคำว่า PROCEDURE หรือ FUNCTION ตามแบบภาษาปาสคาล และให้จบด้วยคำว่า ENDFTN แทน END; ใน IQ0 ไม่มี compound statement ซึ่งจะต้องขึ้นต้นด้วย BEGIN และปิดด้วย END เพราะใน structure statement ได้กำหนดให้มีคำปิดท้ายเฉพาะคือ ENDIF กับ ENDY และสำหรับ structure statement ได้กำหนดไว้เพียง 3 แบบเท่านั้นคือ IF, IF-ELSE และ WHILE ทั้งนี้เพื่อให้โปรแกรมตัวแปรภาษาสั้น

ซีแมนติกของ IQ0

ดังกล่าวแล้วว่า ผมไม่ต้องเขียนคำนิยาม เพื่ออธิบายไวยากรณ์เพราะใช้ของภาษาปาสคาล แต่มีบางเรื่องแตกต่างกันคือ บล็อกแต่ละบล็อกกำหนดเรียกเป็นฟังก์ชัน (Function) ทั้งๆที่ตรงกับโพรซีเดอร์ (Procedure) ในภาษาปาสคาล และไม่ตรงกับฟังก์ชันทั้งในภาษาปาสคาลและซี ไม่มีเหตุผลอะไรหรอกครับ นอกจากเห็นว่าสั้นดี

ฟังก์ชันคือ โปรแกรมย่อยที่ให้ฟังก์ชันอื่นเรียกใช้ โดยที่ฟังก์ชันผู้เรียกจะต้องมีตำแหน่งที่อยู่ในโปรแกรมหลังฟังก์ชันที่ถูกเรียก(ตามแบบภาษาปาสคาล) เรื่องอื่นๆก็เหมือนกันครับที่แตกต่างกันคือ

การกำหนดตัวแปร

ต้องกำหนดไว้ตรงส่วนที่เป็น declaration คือ ระหว่าง heading กับ block แรกเท่านั้น จะกำหนดในส่วนอื่นของโปรแกรมเช่น ในฟังก์ชัน (ตามแบบภาษาปาสคาล) ไม่ได้ (ตลกดี จึงได้ชื่อว่า IQ0 เพราะศูนย์เสียโอกาสที่จะใช้สแต็คเก็บข้อมูลของตัวแปรในฟังก์ชัน ซึ่งจะถูกยกเลิกเมื่อทำงานในฟังก์ชันนั้นเสร็จเรียบร้อยแล้ว เป็นการประหยัดหน่วยความจำ) และจะต้องกำหนดเสมอถึงแม้จะไม่ใช้ (ยิ่งตลกคือ IQ0 หนักขึ้นไปอีก) ส่วนตัวแปรก็มีอยู่เพียง 3 แบบเท่านั้น

การคำนวณ

ดำเนินการได้เฉพาะการ บวก ลบ คูณ และหารเลขจำนวนเต็ม การหารเป็นการหารในแบบปัดเศษทิ้ง และดำเนินการได้เพียงประโยคละ 1 กรรมวิธีเท่านั้น เช่น ต้องการหาเศษของการหารเลข 17 ด้วยเลข 5 หากได้กำหนดตัวแปรไว้เป็น div, mul และ mod จะเขียนโปรแกรมได้ดังนี้

```
div = 17 / 5
mul = div * 5
mod = 17 - mul
```

ซึ่ง mod คือเศษของการหารเลข 2 จำนวนนี้ (จะเห็นว่าเป็น IQ0 มากๆ ครับ) ทั้งตัวหารและตัวตั้งรวมทั้งการดำเนินกรรมวิธีคำนวณอื่นๆ จะเป็นตัวแปรหรือค่าคงตัวก็ได้

เงื่อนไข

เงื่อนไข IF, IF-ELSE และ WHILE จะต้องไม่มีการคำนวณใดๆ (แย่หน่อยครับ)

Standard Command

คือ คำที่ใช้ในโปรแกรมกำหนดไว้เป็นมาตรฐานเพียง 5 คำเท่านั้น คือ ReadInt, WriteInt, ReadChar, WriteChar และ nl เพื่อการอ่าน-เขียนเลขจำนวนเต็ม การอ่าน-เขียนอักขระ และการขึ้นบรรทัดใหม่ตามลำดับ (คำสั่งเดียวกับสตริงจะกำหนดภายหลัง)

ตัวกระทำเปรียบเทียบ

กำหนดไว้เพียง 4 ตัว คือ เท่ากับ มากกว่า น้อยกว่า และไม่เท่ากับ โดยใช้สัญลักษณ์ = > < และ # ตามลำดับ

โปรแกรมตัวแปรภาษา IQ0

หลังจากที่ได้กำหนดไวยากรณ์ของภาษาแล้ว เราก็สามารถนำมาอนุวัติคือ เขียนเป็นโปรแกรมตัวแปรภาษาได้ ซึ่งผมตั้งใจว่า จะเขียนเป็นคอมพายเลอร์แปลซอร์สโค้ดเป็นแมชินโค้ดของซีพียูสมมติเก็บลงไฟล์ แต่ความจริงแมชินโค้ดคือ รหัสกลาง จึงต้องมีโปรแกรมอีกส่วนหนึ่งแปลรหัสกลางเป็นแมชินโค้ดของเครื่องนั้น แต่ผมกลับแปลเป็นภาษาปาสคาล (เพื่อให้ง่ายเพราะเป็นเพียงตัวอย่างเท่านั้น) โปรแกรมส่วนหลังนี้เขียนในแบบอินเทอร์พรีตเตอร์ ซึ่งในการพัฒนาโปรแกรม IQ0 นี้ ผมนำทั้งสองส่วนนี้มารวมกัน

โปรแกรมหลัก (Main)

ในการเขียนโปรแกรมนี้ ผมไม่ต้องวางแนวความคิดในการเขียนโปรแกรมอีก เพราะได้กล่าวไว้ในไวยากรณ์และซีแมนติกเรียบร้อยแล้ว มีเรื่องที่ผมจะกล่าวถึงก่อนคือ ในการเรียงลำดับโปรแกรมผมจะให้เป็นลำดับอักษร A เป็นต้นไป ซึ่งผมจะอ้างถึงโปรแกรมตามนี้และขอให้คุณป้อนเรียงลำดับตามนี้ แต่ในการเรียงลำดับรูปในการพิมพ์ผมต้องการให้ส่วนของโปรแกรมที่กล่าวถึงอยู่ใกล้กับคำอธิบาย รูปจึงไม่เรียงตามลำดับอักษร

ในรูป A เป็นการกำหนดค่าต่างๆ เพื่อใช้ในโปรแกรมเหมือนกับในไมโครฯ 32 มีเปลี่ยนแปลงและเพิ่มเติมบ้างเล็กน้อย ขอให้ดูการกำหนดตัวแปร PRG ย้อนกลับไปครับ จะเห็นว่าเป็นอะเรย์ของข้อมูลแบบ OneCodeT ซึ่งเป็นเรคอร์ด มีฟิลด์อยู่ 2 ฟิลด์ คือ ฟิลด์ ncode สำหรับเก็บรหัสกลาง และฟิลด์ oper สำหรับเก็บแอดเดรสหรือค่าคงตัว ตัวแปร PRG จึงเทียบได้กับโปรแกรมรหัสเครื่องของซีพียูจริง ดังที่เก็บอยู่ในไฟล์ .COM หรือ .EXE ต่อไปหากเก็บลงไฟล์ ผมจะให้นามสกุลเป็น .INT (intermediate)

ตัวโปรแกรมหลักของ IQ0 อยู่ในรูป Q โปรแกรมส่วนนี้เพียงแต่ตรวจสอบว่ามีซอร์สไฟล์ตามชื่อที่ป้อนหรือไม่ หากมีก็จะทำการแปลต่อไป หากไม่มีก็จะแจ้งว่าไม่มีไฟล์นี้

คอมพายล์ (Compile)

โปรแกรมนี้ คือ พระเอกของเรื่องนี้ครับ ได้ใช้ชื่อว่า compile จากไวยากรณ์มีเรื่องที่ผมจะต้องดำเนินการมากมายคือ เขียนโปรแกรมสำหรับแปล statement ต่างๆ แต่ที่ผมจะดำเนินการในฉบับนี้ คือ assignment กับ IF เท่านั้น

ส่วนหัวของโปรแกรม compile อยู่ในรูป B มีการกำหนดค่าประจำและตัวแปรเพื่อใช้ในโปรแกรมส่วนนี้โดยเฉพาะ ตามที่กำหนดในไวยากรณ์ อักขระที่เป็น stop คือใช้คั่นคำ อาจจะดูยาก จึงขอเรียนว่า ประกอบด้วย , ; ( ) วรรค

A

```
PROGRAM   IQ0one(input,output,fi);
CONST  MAXCODE = 1000; MAXMEM = 1000; MAXVAR = 20; MAXFTN = 20;
       MAXREF  = 3; MAXSTD = 5;
TYPE   MnemonicT = String[3];
       NcodeT = (ADDI,SUBI,MULI,DIVI,BEQL,BNEQ,BGTR,BLSS,BGEQ,
                 BLEQ,JUMP,PUSH,PLUS,MINS,JTSR,RETS,INCR,DECR,
                 INNI,INNS,OUTI,OUTS,LODE,STOR,MOVE,HALT,NULL,
                 MVST);
       OneCodeT = RECORD
                    ncode : NcodeT;
                    oper  : Integer;
                  END;
       InstrSetT = ARRAY[NcodeT] OF MnemonicT;
       PRGtype = ARRAY[0..MAXCODE] OF OneCodeT;
       IdenT = String[31];
CONST  InstrSet : InstrSetT =
         ('ADD','SUB','MUL','DIV','BEQ','BNE','BGT','BLT','BGE',
          'BLE','JMP','PSH','PLS','MNS','JSR','RTS','INC','DEC',
          'INI','INS','OTI','OTS','LDE','STO','MOV','HLT','NUL',
          'MVS');
       needOperand : SET OF NcodeT = [BEQL..RETS];
       alphabet : SET OF Char = ['A'..'Z','_'];
       numeric  : SET OF Char = ['0'..'9'];
VAR    srcFi : Text;  finame : String[12];
       PRG : PRGtype;
       error : Boolean;
       Ppntr : Integer;
       mem : ARRAY[0..MAXMEM] OF Integer;
```

Q

```
FUNCTION  canOpen(VAR fi : Text) : Boolean;
BEGIN
  {$I-}  Reset(fi);  {$I+}
  canOpen := (IOresult = 0);
END;


BEGIN   { MAIN }
 Write('Enter source file name.IQ0  '); ReadLn(finame);
 Assign(srcFi, finame);
 IF  canOpen(srcFi)  THEN
   BEGIN
   Compile(srcFi);
{   ListCode(PRG); }
   IF NOT error THEN
     BEGIN
     WriteLn('Program are running..');
     Interpret(PRG); WriteLn;
     WriteLn('  ...end of Program:  ',finame);
     END
   ELSE
     WriteLn('some error detect not run this program');
   END
 ELSE
   WriteLn('Can not open file ',finame);
END.


(*************************************************************)

E>
```

B

```
(*================= COMPILER ================================*)

PROCEDURE  Compile(VAR srcFi : Text);
CONST  MaxWrd = 5;  SPACE = ' ';
       stop : SET OF Char = [',',';','(',')',' ',#13];
       operator : SET OF Char = ['+','-','*','/','=','>','<','$'];
       endWrdSet : ARRAY[1..MaxWrd] OF String[6] =
                   ('ENDPRO','ENDFTN','ENDIF','ELSE','ENDY');
       STDiden : ARRAY[1..MAXSTD] OF String[9] =
                 ('READINT','WRITEINT','READCHAR','WRITECHAR','NL');
VAR
  VARloc : ARRAY[1..MAXVAR] OF
             RECORD
               name : IdenT;
               adr  : Integer;
               END;
  FTNloc : ARRAY[1..MAXFTN] OF
             RECORD
             name : IdenT;
             adr  : Integer;
             para : Integer;
             END;

  Vcntr, Fcntr, memI, Popy  : Integer;
  word : IdenT; lastCh : Char;
  EOfile : Boolean;
```

และรหัส 13 คือ carriage return ตัวแปร VARloc เป็นตัวแปรสำหรับเก็บชื่อตัวแปร และแอดเดรสของหน่วยความจำที่จะเก็บข้อมูลของตัวแปรนี้ สำหรับ FTNloc ก็เช่นกัน แต่สำหรับเก็บชื่อฟังก์ชันกับแอดเดรสต้นของฟังก์ชันนี้ กับจำนวนตัวแปรที่ผ่านมาให้แก่ฟังก์ชันนี้

ตัวโปรแกรมหลักของโปรแกรม Compile อยู่ในรูป N โปรแกรมคอมพายเลอร์ทั่วไป จะอ่านซอร์สโปรแกรมเข้ามาทีละ 1 ประโยค นำมาตรวจสอบและตีความเมื่อถูกต้อง จึงจะแปล แต่ของผมจะอ่านเข้ามาทีละคำมาตรวจสอบและแปลไปพร้อมๆกัน (แต่ที่ทำจริงนั้นจะอ่านทีละ 1 อักขระมารวมเป็นประโยคหรือคำ)

ขอให้ดูการให้ค่าเริ่มต้นแก่ตัวแปลก่อน Ppntr เป็นอินเด็กซ์สำหรับชี้ข้อมูลแต่ละช่องในอะเรย์ PRG ได้ให้ค่าเป็นหนึ่ง เทียบได้กับให้เริ่มโปรแกรมที่แอดเดรสหนึ่ง Vcntr เป็นอินเด็กซ์ชี้ข้อมูลในอะเรย์ VARloc ที่ให้ค่าเป็น MAXREF จะกล่าวในฉบับต่อไป Fcntr เป็นอินเด็กซ์ชี้ข้อมูลในอะเรย์ FTNloc ให้ค่าเริ่มต้นเป็นหนึ่งเพื่อเก็บชื่อและแอดเดรสของฟังก์ชันที่ต้นอะเรย์ ส่วน memI เป็นอินเด็กซ์ชี้อะเรย์ mem ซึ่งสมมติว่าเป็นหน่วยความจำสำหรับเก็บข้อมูล

ตัวโปรแกรม Compile ตามรูป N เริ่มต้นด้วยการอ่านข้อมูลทีละคำ จนกระทั่งถึงคำว่า INT จะเห็นว่าเขียนตามไวยกรณ์ซึ่งได้กำหนดไว้ว่า ส่วน Heading จะมีหรือไม่ก็ได้ และไม่ได้นำมาใช้ประโยชน์ในโปรแกรมจึงอ่านผ่านไป เป็นผลให้ข้อความใดก็ตามที่เขียนไว้ก่อน INT จะถูกอ่านผ่านไปไม่นำมาตีความ

เมื่อถึงคำว่า INT ตามไวยากรณ์จะเป็นตัวแปร จึงให้กำหนดตัวแปรด้วย

```
BEGIN  {Compile}
  Ppntr := 1; Vcntr := MAXREF; Fcntr := 1; memI := -1;
  Popy := 0;
  EOfile := FALSE;  error := FALSE;
  Reset(srcFi);
  REPEAT
    ReadIden(word, lastCh);
  UNTIL  (word = 'INT') OR EOfile;
  AssignVariable(word);
  ReadIden(word, lastCh);
  WHILE (word <> 'ENDPRO') AND (NOT EOfile)  AND (NOT error)  DO
    BEGIN
    DoBLOCK(word,lastCh);
    ReadIden(word, lastCh);
    END;
  Close(srcFi);

WRITELN('Pp=',Ppntr,' Vcntr=',vcntr,' Fcntr=',Fcntr,
        ' memI=',memi,' Popy=',Popy);
END;
```

โปรแกรม AssignVariable เสร็จแล้วให้อ่านคำเข้ามาใหม่ตามไวยากรณ์ จะเป็นบล็อกของฟังก์ชันต่างๆ จึงให้ดำเนินการในบล็อกด้วยโปรแกรม DOBLOCK และให้ทำในวงรอบ WHILE จนกว่าจะจบโปรแกรมคืออ่านพบคำว่า ENDPRO ที่กำหนดไว้ว่าให้ทำงานในวงรอบเมื่อยังไม่จบไฟล์ คือ NOT Eofile ณ ที่นี้และต่อๆไป เป็นมาตรการป้องกันอย่างหนึ่ง หากผู้เขียนโปรแกรมภาษา IQ0 เขียนโปรแกรมผิด เมื่อจบโปรแกรมแล้วให้ปิดไฟล์ จบการทำงานในโปรแกรม Compile คือ จบการคอมพาย

การกำหนดตัวแปร (AssignVariable)

การกำหนดตัวแปรในแบบเลขจำนวนเต็มและอักขระดำเนินการเหมือนกัน แต่ในขั้นนี้ ผมทำเฉพาะเลขจำนวนเต็มก่อน การกำหนดตัวแปรคือ การให้ชื่อตัวแปร (คือคำที่อ่านต่อจาก INT) แก่อะเรย์ VARloc ตรงเรคอร์ดที่ว่าง ตรงฟิลด์ name และให้แอดเดรสของหน่วยความจำที่ว่าง (ในอะเรย์ mem) ตรงฟิลด์ adr ของเรคอร์ดเดียวกัน เสร็จแล้วเพิ่มค่าอินเด็กซ์ที่เกี่ยวข้อง ดังแสดงในโปรแกรม AssignVariable ในรูป D ซึ่งจะทำซ้ำอยู่จนตรวจพบเครื่องหมาย ; ตามที่กำหนดในไวยากรณ์

การดำเนินการใน block (DoBlock)

ในไวยากรณ์เมื่อกำหนดตัวแปรเรียบร้อยแล้ว จะเป็นเรื่องของ block ซึ่งก็คือ ฟังก์ชันหรือคือโปรแกรมย่อยนั่นเอง ส่วนของโปรแกรมต่อไป คือ main ซึ่งเป็น block เหมือนกัน แต่กำหนดให้ชื่อ main และอยู่ท้ายสุด การดำเนินการกับ block ทำด้วยโปร-

```
PROCEDURE  AssignVariable(wwrd : IdenT);
VAR   wrd : IdenT; ch : Char;
BEGIN
  IF  (wwrd = 'INT') OR (wwrd = 'CHAR')  THEN
    BEGIN
      REPEAT
        ReadIden(wrd, ch);
        Vcntr := Vcntr + 1;
        memI  := memI + 1;
        VARloc[Vcntr].name := wrd;
        VARloc[Vcntr].adr  := memI;
      UNTIL  ch = ';' ;
    END;
END;
```

แกรม DoBlock ซึ่งมีส่วนหัวอยู่ที่รูป E และตัวโปรแกรมอยู่ที่รูป M

ตามไวยากรณ์ block จะประกอบด้วย Identifier คือ ชื่อของฟังก์ชัน และ statement กับปิดท้ายด้วยคำว่า ENDFTN แต่ชื่อของฟังก์ชันแบ่งออกเป็น 2 แบบคือ main กับชื่อที่ผู้เขียนโปรแกรมกำหนดเป็นชื่อของโปรแกรมย่อย การดำเนินการกับฟังก์ชันทั้งสองแบบนี้ไม่เหมือนกัน ใน DoBlock จึงแยกออกจากกัน สำหรับ main ได้ให้ค่าแอดเดรสเริ่มต้นในการรันโปรแกรมภาษา IQ0 ด้วยค่า Ppntr ซึ่งในกรณีนี้เมื่อไม่มีโปรแกรมย่อยคำนี้จะเป็น 1 ถ้าเป็นชื่อโปรแกรมย่อยผมได้เว้นไว้ก่อนครับ

ถัดจากชื่อของฟังก์ชันคือ statement ได้มอบให้ DoStatement ดำเนินการ เมื่อกลับมาจะเป็นการปิดท้าย ซึ่งถ้าเป็น main จะเป็นการจบโปรแกรมภาษา IQ0 ถ้าเป็นโปรแกรมย่อยก็จะเป็นการส่งกลับ และเป็นการจบโปรแกรม DoBlock ในการดำเนินการในแต่ละ block (DoBlock ทำงานในวงรอบ WHILE ในโปรแกรม Compile)

กรรมวิธีในแต่ละประโยค (DoStatement)

ตามไวยากรณ์ statement คือ ประโยคมีอยู่ 3 รูปแบบใหญ่ๆ คือ assignment, control และ command ซึ่งโปรแกรม DoStatement ซึ่งมีส่วนหัวอยู่ที่รูป G และตัวโปรแกรมอยู่ที่รูป L เป็นผู้ดำเนินการในเรื่องเหล่านี้ทั้งหมด ขอให้สังเกตด้วยครับว่า โปรแกรมนี้เขียนในรูปแบบของโปรแกรมเรียกตนเอง ทั้งนี้เพราะเราไม่ทราบล่วงหน้าว่า ผู้เขียนโปรแกรมด้วยภาษา IQ0 จะเขียนกี่ประโยคต่อ block

เริ่มต้นด้วยการอ่านคำในภาษา IQ0 1 คำ นำมาตรวจสอบว่าเป็น EndWord เช่น ENDFTN หรือไม่ ถ้าใช่จะเป็นการยุติการเรียกตนเองด้วยการให้ again เป็นเท็จ (จะเห็นต่อไปว่ายุติได้อย่างไร) ถ้าไม่ใช่ให้นำไปตรวจต่อว่า เป็นประโยคของอะไรคือ

- ถ้าเป็นคำมาตรฐานคือ ReadInt, WriteInt, ReadChar, WriteChar หรือ nl ให้ทำ DoSTDiden (ประโยคที่เป็น Command แบ่งออกเป็น Standard คือ 5 คำนี้ กับคำที่ผู้ใช้ตั้งขึ้นเป็นชื่อโปรแกรมย่อย)
- ถ้าเป็น IF ให้ทำ DoIf และอื่น ๆ ดังจะกล่าวต่อไป
- ถ้าเป็นตัวแปร ให้ทำ DoAssignment

E

```
(*---------- DoBLOCK IS MAIN SUBROUTINE OF COMPILER -------------*)
{  Convert BLOCKS OF HIGH LEVEL LANGUAGE (i.e. FUNCTION) into    }
{                    INTERMEDIATE CODE                           }

PROCEDURE  DoBLOCK(Bwrd : IdenT; Blch :char);
VAR     again : Boolean;
        lastStop, FTNname : IdenT;
        dist : Integer;
```

M

```
BEGIN   { DoBLOCK }
  Popy := 0;
  FTNname := Bwrd;
  IF  Bwrd = 'MAIN'  THEN
    MarkMain
  ELSE
    begin  end;
  again := TRUE;
  DoStatement;
  IF  FTNname = 'MAIN'  THEN
    MarkEnd
  ELSE
    begin  end;
END;
```

G

```
  (*========== DoStatement IS MAIN FTN OF DoBLOCK ==========*)

PROCEDURE  DoStatement;
VAR    swrd : IdenT; slCh : Char;
       parm, varT, PCy, PCif, idx, addr : Integer;
```

L

```
BEGIN   { DoSTATEMENT }
  ReadIden(swrd,slch);
  IF  EndWord(swrd) OR error  THEN
    BEGIN lastStop := swrd; again := FALSE  END
  ELSE
    BEGIN
    IF  isSTDiden(swrd,idx)  THEN  DoSTDiden(swrd,idx)
    ELSE IF  swrd = 'IF'  THEN  DoIf
    ELSE IF IsVar(swrd,addr,varT)  THEN  DoAssignment(addr,varT)
         ELSE
           BEGIN
           error := TRUE;
           WriteLn(' ERROR: can not recognize ',swrd);
           END;
    END;
  IF  again  THEN  DoStatement;
END;
```

- หากไม่ตรงกับประโยคแบบใดเลยจะแจ้งว่าผิด

เสร็จแล้วจะตรวจดูว่า again เป็นจริงหรือเท็จ หากเป็นจริงก็จะไปเริ่มต้นโปรแกรมนี้ใหม่ด้วยการเรียกตนเอง อ่านคำเข้ามาใหม่ตรวจสอบใหม่จนพบคำว่า ENDFTN ให้ again เป็นเท็จ ดังนั้นเมื่อมาตรวจสอบตอนท้ายของโปรแกรมก็จะไม่เรียกตนเองอีกเป็นการยุติการเรียกตนเอง ดำเนินการส่งกลับและจบจากโปรแกรมนี้กลับไป DoBlock

ประโยคให้ค่าแก่ตัวแปร (DoAssignment)

ประโยคที่ให้ค่าแก่ตัวแปร assignment statement ตามไวยากรณ์จะประกอบด้วย identifier และเครื่องหมาย เท่ากับ และ expression และมีเครื่องหมาย ; ปิดท้าย ซึ่งจะแจงรูปแบบได้ดังนี้

ตัวแปร = ค่าคงตัว;
ตัวแปร = ตัวแปร;
ตัวแปร = ค่าคงตัว + ค่าคงตัว;
ตัวแปร = ค่าคงตัว + ตัวแปร;
ตัวแปร = ตัวแปร + ค่าคงตัว;
ตัวแปร = ตัวแปร + ตัวแปร;

ทั้งนี้โดยให้เครื่องหมาย + แทนเครื่องหมายในการคำนวณทั้งหมด จากรูปแบบทั้งหกนี้ หากให้ A เป็นตัวแปร x และ y เป็นตัวแปรหรือค่าคงตัวก็ได้ จะยุบเหลือเพียง 2 รูปแบบ แต่ผมมีเรื่องที่จะเสริมคือ ไวยากรณ์ของภาษา IQ0 ได้กำหนดให้ปิดท้ายประโยคด้วย ; โดยไม่มีวรรค (ต่างกับภาษาปาสคาลที่มีหรือไม่มีก็ได้) ทั้งนี้เพื่อให้เขียนโปรแกรม compile ได้สั้น แต่ถ้าหากให้มีหรือไม่มีก็ได้ รูปแบบของประโยค assignment จะมี 4 รูปแบบคือ

A = x;
A = x ;
A = x + y;
A = x + y ;

ในการเขียนโปรแกรม เราจึงต้องคำนึงว่า จะต้องพบรูปแบบหนึ่งรูปแบบใดใน 4 รูปแบบนี้ ซึ่งผมจะเขียนตามนี้เพื่อเป็นตัวอย่าง มีอีกเรื่องหนึ่ง ซึ่งเห็นว่าสมควรจะกล่าวในตอนนี้คือ ในการเขียนโปรแกรมต่อจากนี้ จะต้องคำนึงถึงสถาปัตยกรรมภายในและชุดคำสั่งของซีพียูที่จะนำภาษา IQ0 ไปใช้ ซึ่งในกรณีนี้คือ ซีพียูสมมติและรหัสกลาง เช่นจะเขียนโปรแกรมภาษาเครื่องให้ทำประโยค

A = B + 10;

จะเขียนได้ดังนี้

PUSH A ; แอดเดรสของ A
PUSH B ; แอดเดรสของ B
MOV ; ข้อมูลของ B
PUSH 10 ; ค่าคงตัว
ADDI ; บวกกัน
STOR ; ให้ค่าแก่ A

โปรแกรมที่ดำเนินการในเรื่องนี้คือ DoAssignment รูป J เริ่มด้วยเก็บค่า Popy (จะกล่าวภายหลัง) แล้วเก็บแอดเดรสของตัวแปรซึ่ง DoStatement ผ่านมาให้ลงสแต็ค อ่านเครื่องหมายเท่ากับหากไม่ใช่จะให้ข้อความผิดพลาด (อยู่ตอนท้ายของโปรแกรมนี้) หากใช่เครื่องหมายเท่ากับให้อ่านตัวตั้งด้วย ReadOper แล้วให้ PushOperand ดำเนินการเก็บลงสแต็ค ตรงนี้ผมไม่ได้ตรวจว่ามีตัวตั้งหรือไม่ ซึ่งความจริงควรตรวจ

เสร็จแล้วตรวจอักขระที่อยู่ติดกับตัวตั้ง (ซึ่งมีค่าอยู่ใน ch) ว่าเป็น ; หรือไม่ ถ้าใช่ก็จบประโยคนี้ ไม่มีการคำนวณคงให้ค่าตัวตั้งแก่ตัวแปรได้เลยด้วย StoreResult แต่ในโปรแกรมเขียนเป็นเมื่อไม่ใช่ ; ให้ ซึ่งตรงนี้อาจเป็นวรรคตามด้วย ; หรือเป็นเครื่องหมายในการคำนวณ หรือผู้เขียนโปรแกรมลืมใส่เครื่องหมาย ; หรือลืมเครื่องหมายคำนวณ หรืออาจเป็นความผิดอื่นๆ

จึงนำมาตรวจว่าเป็นเครื่องหมายในการคำนวณหรือไม่ หากใช่ให้อ่านตัวกระทำเช่นตัวบวกแล้วเก็บลงสแต็ค

```
PROCEDURE  DoAssignment(adr, vt : Integer);
VAR  ch,opr : Char; wrd : IdenT;
     conI,tem,dif : Integer;
BEGIN
  tem := Popy;
  PushAddress(adr,vt);        { push address of variable }
  ReadSkip(opr);              { read assign operator = }
  IF opr = '=' THEN
    BEGIN
    ReadOper(wrd,ch);  PushOperand(wrd);  { 1st operand }
    IF  ch <> ';'  THEN
      BEGIN
      ReadSkip(opr);
      IF  opr IN ['+','-','*','/']  THEN
        BEGIN
        ReadOper(wrd,CH); PushOperand(wrd);  { 2nd operand }
        DoCalculate(opr);
        END
      ELSE
        IF  opr <> ';'  THEN
          BEGIN
          error := TRUE;
          Writeln(' ERROR: ',opr,' is not operator');
          END;    { if <>, if IN }
      END;        { if <>  }
      dif := Popy - tem;
      StoreResult(dif);
    END           { if =  }
  ELSE
    BEGIN
    error := TRUE;
    WriteLn(' ERROR: ',opr,' is not assign operator');
    END;
END;
```

ดำเนินการคำนวณ แล้วไปให้ค่าแก่ตัวแปร

หากไม่ใช่เครื่องหมายในการคำนวณ ก็อาจเป็นประเด็นแรกคือ วรรคตามด้วย ; ซึ่งขณะนี้ตัวแปร opr เก็บค่านี้อยู่ จึงนำมาตรวจว่าใช่หรือไม่ ถ้าใช่ก็ไปให้ค่าแก่ตัวแปร ถ้าไม่ใช่คือประเด็นหลังเป็นข้อผิดพลาด จึงให้พิมพ์ข้อความถึงข้อผิดพลาด ส่วนท้ายของโปรแกรมนี้คือข้อความถึงข้อผิดพลาดในเรื่องเครื่องหมายเท่ากับดังกล่าวแล้ว โปรแกรมย่อยนี้เรียกใช้โปรแกรมย่อยอื่นอยู่หลายตัว ซึ่งผมขอนำไปรวมกล่าวในตอนท้าย (อยู่ในฉบับต่อไป)

ประโยค IF

ประโยค control มีอยู่ 3 แบบ คือ IF, IF-ELSE และ WHILE ในฉบับนี้ ผมขอกล่าวแต่เพียง IF อันเดียวก่อน ตามไวยากรณ์ได้กำหนดรูปแบบของ IF ไว้เป็น

```
IF  เงื่อนไขเป็นจริง THEN
      statement;
      "
      statement;
ENDIF
```

คือเมื่อเงื่อนไขเป็นจริง จะทำประโยคที่อยู่ระหว่าง THEN กับ END-IF ซึ่งจะมีกี่ประโยคก็ได้และจะเป็นประโยคแบบใดก็ได้ หากเป็นเท็จจะข้ามประโยคเหล่านี้ไป ขอให้ดูตัวอย่างโปรแกรมที่เขียนด้วยรหัสกลาง ตามโปรแกรมที่เขียนด้วยภาษา IQ0 ดังนี้

```
IF  A < B THEN
      A = A - B;
ENDIF
```

จะเขียนเป็นโปรแกรมรหัสกลางได้ดังนี้

```
IF:       PUSH A
          MOVE
          PUSH B
          MOV
          BLEQ ENDIF
THEN :    -    -
          -    -
          STOR
ENDIF:
```

```
(*........ Major subroutine in DoStatement ..............*)

FUNCTION  EndWord(wrd : IdenT) : Boolean;
VAR   i : Integer;
BEGIN
  EndWord := FALSE;
  FOR  i := 1 TO MaxWrd  DO
    IF  wrd = endWrdSet[i]  THEN  EndWord := TRUE;
END;


PROCEDURE  DoIfHead(resWrd : IdenT);
VAR  wrd : IdenT; opr, lch : Char; adr : Integer;
BEGIN
  ReadIden(wrd, lch);      { first operand var }
  IF  IsVar(wrd, adr, varT)  THEN
    BEGIN
    PushVariable(adr,varT);          { data of first operand }
    ReadSkip(opr);          { relational operator }
    ReadOper(wrd,lch);      { second operand }
    PushOperand(wrd);       { data of second operand }
    BranchOnCond(opr);      { compare and branch on condition }
    ReadIden(wrd, lch);
    IF  wrd <> resWrd  THEN
      BEGIN
      error := TRUE;
      WriteLn(' ERROR: IF without THEN.. ',wrd)
      END;
    END
  ELSE
    BEGIN
    error := TRUE;
    WriteLn(' ERROR: IF without variable.. ',wrd)
    END;
END;

PROCEDURE  DoIf;
BEGIN
  DoIfHead('THEN'); PCif := Ppntr - 1;
  DoStatement;

  DoEndIf(PCif); again := TRUE;
END;
```

คำว่า IF: , THEN: และ ENDIF: เป็นที่หมายในการกระโดด ซึ่ง IF: และ THEN: ไม่จำเป็นต้องมี เขียนไว้เพื่อให้เทียบเคียงกันเท่านั้น ส่วนที่ละไว้คือการคำนวณและให้ค่าแก่ A คล้ายกับที่ได้เคยยกเป็นตัวอย่าง ขอให้สังเกตว่าต้องกระโดดไปที่ ENDIF: เมื่อ A น้อยกว่าหรือเท่ากับ B คือถ้า A มากกว่า B จึงจะทำคำสั่งตรง THEN: และที่ตามมาจนถึง STOR คือก่อน ENDIF:

การทำงานในโครงสร้าง IF จึงแบ่งเป็น 3 ตอนคือ ตอนที่หนึ่ง เริ่มจาก IF: ไปถึงก่อน THEN: ตอนที่สองตั้งแต่ THEN: ไปถึงก่อน ENDIF: ซึ่งจะเป็น statement ตามที่ผู้เขียนโปรแกรม IQ0 จะกำหนด ตอนที่สามเป็นการดำเนินการต่างๆ ตรง ENDIF:

ขอให้ดูโปรแกรม DoIf ในรูป I การทำงานในตอนหนึ่งได้มอบให้ DoIf Head รูป I โปรแกรมนี้จะทำตรงส่วนหัวของโครงสร้าง IF เช่น IF A>B ซึ่งดำเนินการตรงๆคือ อ่านตัวแปรมาเก็บค่าลงสแต็คอ่านตัวดำเนินการ (operator) และก็อ่านค่าที่จะนำมาเทียบ (อาจเป็นตัวแปรหรือค่าคงตัว) เก็บลงสแต็ค เสร็จแล้วให้กระโดดตรงข้ามเงื่อนไขที่กำหนดไว้ (ดังที่ยกมาเป็นตัวอย่าง) ด้วยการเรียกใช้ BranchOnCond แต่ตอนนี้ยังไม่ได้

ให้แอดเดรสที่จะกระโดดไป หากดูในโปรแกรม DoIf จะเห็นว่า ได้เก็บแอดเดรสตรงที่ให้กระโดดนี้ไว้ในตัวแปร PCif

ท่อนต่อไปของ DoIfHead เป็นการตรวจสอบว่าผู้เขียนโปรแกรม IQ0 เขียนผิดไวยากรณ์หรือไม่คือ ไม่มีคำว่า THEN หรือไม่ได้ให้ตัวแรกเป็นตัวแปร (น่าจะตรวจตัวดำเนินการด้วย แต่ก็ไม่ได้ตรวจ) เสร็จแล้วกลับไป DoIf เก็บค่า Ppntr ดังกล่าวมาแล้ว

ตอนที่สอง เป็นการนำ statement ตามแต่ผู้ใช้จะกำหนดด้วยการเรียกตนเอง สมมติว่ามี 2 ประโยค ก็จะเรียกตนเองซ้ำอยู่ 2 ครั้ง ในแต่ละครั้งอาจจะเป็น command หรือ control หรือ assignment ก็ได้ เมื่อเรียกตนเองซ้ำอยู่ 2 ครั้งแล้ว ก็จะมาทำ IF again ในรูป L ซึ่งจะเป็นการเรียนตนเองต่ออีก 1 ครั้ง

ซึ่งในภาษา IQ0 จะกำหนดให้ปิดท้ายโครงสร้าง IF ด้วย ENDIF จึงกำหนดให้เป็นพวก EndWord ด้วย ทำให้ again เป็นเท็จ ไม่เรียกตนเอง ดำเนินการส่งกลับมายังตำบลที่เรียกตนครั้งสุดท้ายในโปรแกรม DoIf คือที่ผมทำให้บรรทัดว่างไว้แล้วทำ DoEndIf

โปรแกรมย่อย DoEndIf คือตอนที่สามของโครงสร้าง IF โปรแกรมนี้อยู่ในรูป F2 จะให้แอดเดรส ณ จุดนี้แก่คำสั่งกระโดด ที่กล่าวในตอนที่หนึ่ง และยกเลิกค่าในสแต็ค กลับไป DoIf ได้ให้ again เป็นจริงใหม่เพื่อให้เรียกตนเองต่อไปจนกว่าจะครบทุกประโยคที่เขียนไว้ใน block (ขอให้นึกถึงว่าเป็นไปตามรูปแบบของภาษาปาสคาลครับ)

คำมาตรฐาน (isSTDiden กับ DoSTDiden)

ดังที่กล่าวแล้วว่า ได้กำหนดคำมาตรฐาน (Standard Identifier) ไว้ 5 คำ คือ ReadInt, WriteInt, ReadChar, Write Char และ nl ในโปรแกรม DoStatement ได้ตรวจว่าเป็นคำเหล่านี้หรือไม่ ด้วยการเรียกใช้ฟังก์ชัน isSTDiden รูป K ซึ่งถ้าใช่จะให้ isSTDiden เป็นจริง และให้ค่าคำเป็น 1 ถึง 5 กลับมาในตัวแปร idx

แล้วให้ทำ DoSTDiden อยู่ในรูป K เช่นกัน นำมาตรวจดูว่า ถ้าเป็น 5 ให้ทำ NewLine คือขึ้นบรรทัดใหม่ นอกนั้นให้อ่านค่าตัวแปรในวงเล็บมาเก็บลงสแต็ค แล้วแยกไปทำ อ่าน หรือเขียนกับตัวเลขหรือตัวอักษร ดังที่เห็นในโปรแกรม

จะเห็นว่า การออกแบบคำมาตรฐานเป็นไปในแบบของเทอร์โบปาสคาล ซึ่งถ้าไม่มีซอร์สโปรแกรมจะเพิ่มคำไม่ได้เลย แตกต่างกับบางระบบเช่นของไมโครซอฟต์ทำเป็นไลบรารี่สามารถเพิ่มคำในไลบรารี่ได้

โปรแกรมสนับสนุน

โปรแกรมดังกล่าวมา ต้องใช้โปรแกรมย่อยสนับสนุนจำนวนหนึ่ง ซึ่งผมได้แบ่งออกเป็น 4 พวก คือ

เกี่ยวกับการอ่านไฟล์อยู่ในรูป C มีหน้าที่การทำงานดังนี้

. **ReadCh (ch)** อ่านข้อมูลจากซอร์สไฟล์ทีละอักขระ นำมาแปลงอักษรเล็กเป็นอักษรใหญ่ตรวจว่า ถ้าเป็นอักขระ #13 คือ Line feed ให้ค่าเป็นวรรค และให้ขึ้นบรรทัดใหม่ เสร็จแล้วให้แสดงอักขระนี้ออกจอภาพ เพื่อการตรวจหาที่ผิด นอกจากนี้ หากตรวจพบว่าจบไฟล์จะให้ EoFile เป็นจริง ให้ค่าอักขระกลับไปในตัวแปร ch

. **ReadWord (wrd, ch)** อ่านข้อมูลทีละคำ โดยเรียกใช้ ReadCh จนตรวจพบอักขระที่เป็น stop ให้ค่าคำที่อ่านได้กลับไปใน wrd และอักขระที่เป็น stop ใน ch

. **ReadIden (wrd, ch)** อ่านข้อมูลที่ขึ้นต้นด้วยตัวอักษรทีละคำ โดยเรียกใช้ ReadCh อ่านข้อมูลผ่านอัก-

K

```
FUNCTION isSTDiden(swrd : IdenT; VAR idx : Integer) : Boolean;
VAR   id : Integer;
BEGIN
  isSTDiden := TRUE;
  FOR  id := 1 TO MAXSTD  DO
    IF  swrd = STDiden[id]  THEN
      BEGIN IDX := ID; exit END;
  isSTDiden := FALSE;
END;

PROCEDURE  DoSTDiden(wrd : IdenT; idx : Integer);
VAR  pssvar : IdenT; ch : Char; adr : Integer;
BEGIN
  IF  idx = 5  THEN  NewLine
  ELSE
    BEGIN
    ReadIden(pssvar,ch);
    IF  IsVar(pssvar,adr,varT) \THEN
      BEGIN
      PushAddress(adr,varT);
      CASE  idx  OF
       1 : MLrdeInt;
       2 : MLwrtInt;
       3 : MLrdeChar;
       4 : MLwrtChar;
      END;    { case }
      END     { IF }
    ELSE
      WriteLn(' ERROR: ',wrd,' RWvar must be variable');
    END;  { if, if }
END;  { ifStdIdenDo }
```

```
PROCEDURE  ReadCh(VAR ch : Char);
BEGIN
  Read(srcFi, ch);  ch := UpCase(ch);
  IF  ch = #13  THEN
    BEGIN  ch :=SPACE; ReadLn(srcFi); WriteLn  END;
  IF  EOF(srcFi)  THEN  EOFile := TRUE;
  Write(ch);
END;

PROCEDURE  ReadWord(VAR wrd : IdenT; VAR ch : Char);
VAR  i : Integer;
BEGIN
  i := 0;
  WHILE  (NOT (ch IN stop))  AND NOT EOfile  DO
    BEGIN
    i := i + 1;  wrd[i] := ch;
    ReadCh(ch)
    END;
  wrd[0] := Chr(i);
END;


PROCEDURE  ReadIden(VAR wrd : IdenT; VAR lch : Char);
BEGIN
  REPEAT
    ReadCh(lch);
  UNTIL (lch IN alphabet) OR EOfile;
  IF  NOT EOfile  THEN  ReadWord(wrd,lch);
END;


PROCEDURE  ReadOper(VAR wrd : IdenT; VAR ch : Char);
BEGIN
  REPEAT
    ReadCh(ch);
  UNTIL  (ch IN alphabet) OR (ch IN numeric) OR EOfile;
  IF  NOT EOfile  THEN  ReadWord(wrd,ch);
END;


PROCEDURE  ReadSkip(VAR ch : Char);
BEGIN
  REPEAT
    ReadCh(ch)
  UNTIL  (ch <> SPACE) OR EOfile;
END;
```

ขระต่าง ๆ จนกว่าจะถึงอักขระที่เป็นอักษร เรียกใช้ ReadWord เพื่ออ่านเป็นคำส่งกลับใน wrd และ stop ใน ch

. ReadOpr (wrd, ch) คล้ายกับ ReadIden แต่อ่านคำที่ขึ้นต้นด้วยอักษรหรือตัวเลข จุดประสงค์เพื่อใช้ในตำแหน่งที่อาจเป็นตัวแปรหรือค่าคงตัว

เกี่ยวกับการกำหนดตัวแปร อยู่ในรูป D ขณะนี้มีเพียงโปรแกรมเดียวคือ

. AssignVar (wwrd) โปรแกรม Compile จะผ่านคำว่า INT หรือ CHAR มาให้ AssignVar เป็นการแจ้งให้ทราบว่า ตัวแปรที่ตามมาเป็นเลขจำนวนหรืออักขระ ซึ่ง AssignVar จะอ่านชื่อตัวแปรทีละตัวมาเก็บในฟิลด์ name และให้แอดเดรสของหน่วยความจำที่กำหนดให้เป็นของตัวแปรตัวนี้ในฟิลด์ adr ทำซ้ำอยู่จนกว่าจะพบเครื่องหมาย ;

โปรแกรมใช้งานทั่วไป อยู่ในรูป H1 และ H2 เป็นโปรแกรมสนับสนุนของโปรแกรม DoStatement ได้แก่

. IsVar (wrd, adr) เป็นฟังก์ชันโดยที่ผู้เรียกจะต้องผ่านชื่อตัวแปรมาใน wrd ให้ IsVar ตรวจหาในอะเรย์ VARloc ว่ามีชื่อนี้ตามที่ AssignVar บันทึกไว้หรือไม่ หากพบจะทำให้ IsVar เป็นจริง และให้แอดเดรสของข้อมูลของตัวแปรนี้ส่งกลับไปใน adr

. PushOper (wrd) ในกรณีที่เราต้องการเก็บค่าข้อมูลลงสแต็ค โดยที่ค่านี้อาจเป็นตัวแปรหรือค่าคงตัวก็ได้ เราจะเรียกใช้ PushOper ซึ่ง PushOper จะนำมาตรวจสอบว่า ถ้าเป็นตัวแปรก็

```
(*.......... GENERAL FTN USE IN DoSTATEMENT ..........*)


FUNCTION  IsVar(wrd : IdenT; VAR adr, vt : Integer) : Boolean;
BEGIN
  vt := 0;
  REPEAT
    vt := vt + 1;
  UNTIL (wrd = VARloc[vt].name) OR (vt > Vcntr);
  IF  wrd = VARloc[vt].name  THEN
    BEGIN  IsVar := TRUE; adr := VARloc[vt].adr  END
  ELSE  IsVar :=FALSE
END;

PROCEDURE  PushAddress(adr,vt : Integer);
BEGIN
  IF  vt > MAXREF  THEN  PushMemAdr(adr)
  ELSE  begin  end;
END;

PROCEDURE  PushVariable(adr,vt : Integer);
BEGIN
  IF  vt > MAXREF  THEN  PushMemVar(adr)
  ELSE  begin  end;
END;

PROCEDURE  PushOperand(wrd : IdenT);
VAR  adr, conI, res, vt : Integer;
BEGIN
  IF IsVar(wrd, adr, vt)  THEN
    PushVariable(adr,vt)
  ELSE
    BEGIN
    Val(wrd, conI, res);
    IF  res = 0  THEN
      PushConst(conI)
    ELSE
      BEGIN
      error := TRUE;
      WriteLn(' ERROR: ',conI,' is not constant');
      conI := 0; PushConst(conI);
      END;
    END;
END;
```

จะเรียกใช้ PushVar ถ้าไม่ใช่ให้พิจารณาว่า เป็นค่าคงตัวก่อน แต่ข้อมูลในขณะนี้ยังไม่ใช่เลขจำนวนหากเป็นสตริงของตัวเลข จึงต้องเปลี่ยนเป็นเลขจำนวนด้วย Val ของเทอร์โบปาสคาล หากเปลี่ยนได้ถูกต้องก็จะเก็บลงสแต็คด้วย PushConst หากเปลี่ยนไม่ได้จะให้ข้อความบ่งถึงข้อผิดพลาด

. **BranchOnCord (ch)** ดำเนินการกระโดดโดยแปลงเงื่อนไขของการกระโดด จากเครื่องหมายเป็นรหัสกลาง เช่น เครื่องหมาย = เป็น BNEQ (ให้กระโดดเมื่อไม่เท่ากันตามที่ยกตัวอย่างใน IF) ได้รหัสกลางใน mne และเรียกใช้ MLbrance อีกทอดหนึ่ง

. **DoCalculate (ch)** ดำเนินการคำนวณโดยแปลงเครื่องหมายในการคำนวณเป็นรหัสกลาง เช่น + เป็น ADDI แล้วเรียกใช้ MLcalculate อีกทอดหนึ่ง

ส่วนสร้างโปรแกรมรหัสกลาง จากการดำเนินการที่ผ่านมาทั้งหมด โปรแกรมในส่วนนี้จะเป็นโปรแกรมที่แปลโปรแกรมในภาษาสูงเป็นโปรแกรมรหัสกลางอย่างแท้จริง เทียบได้กับการสร้างโปรแกรมภาษาเครื่อง การดำเนินการโดยส่วนรวมคือ นำรหัสกลางไปเก็บในอะเรย์ PRG ตรงช่อง Ppntr (ได้ให้ค่าเป็น 1 เมื่อเริ่มต้น) ที่ฟิลด์ ncode และให้ค่าแอดเดรสหรือค่าคงตัวหากมีในฟิลด์ oper แล้วเพิ่มค่า Ppntr ขึ้นหนึ่ง (ในการอ่านคำอธิบายโปรแกรมตอนนี้ กรุณานำรูป H ในเรื่องภาษาเครื่องสำหรับภาษาสูง ไมโครฯ ฉบับที่ 32 มาประกอบการอ่านด้วยครับ) โปรแกรมในส่วนนี้แบ่งออกเป็น 3 ส่วนย่อยคือ

* **เงื่อนไขและการให้ค่า** อยู่ในรูป F1 ประกอบด้วย

. **PushAdr (adr)** เพื่อเก็บค่า adr ลงสแต็ค (ที่ชั้นบนสุด) โดยให้รหัส PUSH กับ adr (แก่ PRG[Ppntr])

. **PushVar (adr)** เพื่อเก็บค่าข้อมูลของตัวแปรที่ผ่านค่า adr มาให้ลงสแต็ค โดยเรียกใช้ PushAdr แล้วให้รหัส MOVE

. **PushConst (conI)** เพื่อเก็บค่าคงตัวลงสแต็ค โดยให้รหัส PUSH กับ conI (ความจริงคือดำเนินการเช่นเดียวกับ PushAdr ที่แยกกันเพื่อให้อ่านโปรแกรมสะดวก)

H2

```
PROCEDURE  BranchOnCond(ch : Char);
VAR  mne : NcodeT;
BEGIN
  CASE  ch  OF
   '=' : mne := BNEQ;
   '>' : mne := BLEQ;
   '<' : mne := BGEQ;
   '#' : mne := BEQL;
  ELSE
    mne := BEQL;
    error := TRUE;
    WriteLn(' ERROR: ',InstrSet[mne],
            ' is not relational operator');
  END;  {case}
  MLbranch(mne);
END;

PROCEDURE  DoCalculate(ch : Char);
Var  mne : NcodeT;
BEGIN
  CASE  ch  OF
   '+' : mne := ADDI;
   '-' : mne := SUBI;
   '*' : mne := MULI;
   '/' : mne := DIVI;
  END;  {case}
  MLcalculate(mne);
END;
```

F1

```
(*................. TRANSLATE PART .......................*)

PROCEDURE  PushMemAdr(addr : Integer); { push address of var }
BEGIN                                  {  to top of stack  }
  PRG[Ppntr].ncode := PUSH; Popy := Popy + 1;
  PRG[Ppntr].oper  := addr; Ppntr := Ppntr + 1;
END;


PROCEDURE  PushMemVar(addr : Integer);    { push data specify by }
BEGIN                                { addr of var to top of stack }
  PushMemAdr(addr);
  PRG[Ppntr].ncode := MOVE; Ppntr := Ppntr + 1;
END;


PROCEDURE  PushConst(conI : Integer);  { push const to top of }
BEGIN                                  {   STACK }
  PRG[Ppntr].ncode := PUSH; Popy := Popy + 1;
  PRG[Ppntr].oper  := conI; Ppntr := Ppntr + 1;
END;

PROCEDURE  MLbranch(mne : NcodeT);  { branch on not cond. }
BEGIN
  PRG[Ppntr].ncode := mne;
  Ppntr := Ppntr + 1;
END;

PROCEDURE  MLcalculate(mne : NcodeT);  { t-1 operator t }
BEGIN
  PRG[Ppntr].ncode := mne; Popy := Popy - 1;
  Ppntr := Ppntr + 1;
END;

PROCEDURE  StoreResult(dif : Integer);
BEGIN
  PRG[Ppntr].ncode := STOR; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := MINS;
  PRG[Ppntr].oper  := dif - 1; Ppntr := Ppntr + 1;
  Popy := Popy - dif
END;
```

. **MLbranch (mne)** เพื่อดำเนินการกระโดดโดยให้รหัสในการกระโดดซึ่งอยู่ในตัวแปร mne แต่ไม่ได้ให้แอดเดรส (ดังที่กล่าวในเรื่อง IF)

. **MLCalculate (mne)** เพื่อดำเนินการคำนวณโดยให้รหัสในการคำนวณซึ่งอยู่ในตัวแปร mne

. **StorResult** เพื่อให้ค่าแก่ตัวแปรโดยให้รหัส STOR ซึ่งการให้ค่าแก่ตัวแปรนั้น โปรแกรมที่ดำเนินการในเรื่องนี้ จะเก็บค่าแอดเดรสของตัวแปรลงสแต็คก่อน แล้วเก็บค่าที่จะให้วางทับลงไป ซึ่งอาจจะเป็น ค่าคงตัว ตัวแปร หรือผลจากการคำนวณ (ดังกล่าวในประโยคให้ค่าแก่ตัวแปร) การดำเนินการในแต่ละขั้นตอนนี้ทำให้ SP เปลี่ยนแปลงค่าไม่เท่ากัน จึงได้ติดตามการเปลี่ยนแปลงนี้ด้วยตัวแปร dif เมื่อจบกรรมวิธีเราจะต้องยกเลิกค่าในสแต็คด้วยการทำให้ SP มีค่าเดิม คือค่าก่อนที่จะทำ assignment statement โดยการลบค่า SP ด้วยค่า dif แต่ในโปรแกรมนี้ รหัส STOR จะทำให้ค่า SP ลดลงหนึ่งอยู่แล้วการดำเนินการจึงทำโดยให้รหัส MINS กับ dif-1

*** การดำเนินการกับบล็อก** อยู่ในรูป F2 คำว่า บล็อก ในที่นี้หมายรวมถึง block ของ program และบล็อกของ control statement ประกอบด้วย

.DoEndIf เพื่อให้ค่าแอดเดรสแก่คำสั่งกระโดดที่อยู่ตอนหัวของ IF มายัง ENDIF โดยให้แอดเดรส ณ Ppntr ปัจจุบันแค่ PRG [PCif].adr เสร็จแล้วลดค่า SP ลง 2 โดยให้รหัส MINS กับ 2 (ค่า 2 คือค่าที่ SP เพิ่มขึ้นเนื่องจากการเปรียบเทียบค่า)

. **MarkEnd** เพื่อแสดงว่าจบโปรแกรมทำให้โปรแกรม Interpret หยุดการตีความรหัสกลาง ทำโดยให้รหัส HALT

*** คำมาตรฐาน** อยู่ในรูป F3 เป็นโปรแกรมที่ดำเนินการเกี่ยวกับคำมาตรฐานโดยที่ได้กำหนดไว้ในหลักภาษา IQ0 ว่าการผ่านค่าทำได้แบบเดียวคือ pass by reference ซึ่งจะต้องผ่านค่าแอดเดรสของตัวแปรมาทางสแต็ค การดำเนินการ

- Write จึงประกอบด้วยรหัส MOVE กับ OUTI หรือ OUTS เท่านั้น

- Read จึงประกอบด้วยรหัส INNI หรือ INNS กับ STOR และการยกเลิกค่าในสแต็คโดยรหัส MINS กับ 1 เท่านั้น

F2

```
PROCEDURE DoEndIf(oldPC : Integer);
BEGIN
  PRG[oldPC].oper := Ppntr;
  PRG[Ppntr].ncode := MINS;
  PRG[Ppntr].oper  := 2;    Ppntr := Ppntr + 1;
  Popy := Popy - 2;
END;


PROCEDURE Call(adr : Integer);
BEGIN
  PRG[Ppntr].ncode := JTSR;
  PRG[Ppntr].oper  := adr; Ppntr := Ppntr + 1;
  Popy := Popy + 1;
END;

PROCEDURE MarkEnd;
BEGIN
  PRG[Ppntr].ncode := HALT
END;

PROCEDURE MarkMain;
BEGIN
  PRG[0].ncode := JUMP;
  PRG[0].oper  := Ppntr;
END;
```

F3

```
PROCEDURE MLwrtInt;
BEGIN
  PRG[Ppntr].ncode := MOVE; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := OUTI; Ppntr := Ppntr + 1;
  Popy := Popy - 1;
END;

PROCEDURE MLwrtChar;
BEGIN
  PRG[Ppntr].ncode := MOVE; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := OUTS; Ppntr := Ppntr + 1;
  Popy := Popy - 1;
END;

PROCEDURE MLrdeInt;
BEGIN
  PRG[Ppntr].ncode := INNI; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := STOR; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := MINS;
  PRG[Ppntr].oper  := 1;    Ppntr := Ppntr + 1;
  Popy := Popy - 1;
END;

PROCEDURE MLrdeChar;
BEGIN
  PRG[Ppntr].ncode := INNS; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := STOR; Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := MINS;
  PRG[Ppntr].oper  := 1;    Ppntr := Ppntr + 1;
  Popy := Popy - 1;
END;


PROCEDURE newLine;
BEGIN
  PRG[Ppntr].ncode := PUSH;
  PRG[Ppntr].oper  := 13;    Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := OUTS;  Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := PUSH;
  PRG[Ppntr].oper  := 10;    Ppntr := Ppntr + 1;
  PRG[Ppntr].ncode := OUTS;  Ppntr := Ppntr + 1;
END;
```

อินเทอร์เพรทเตอร์ (Interpret)

ดังกล่าวแล้วว่าโปรแกรมตัวแปรภาษา IQ0 แบ่งเป็น คอมพายเลอร์และอินเทอร์เพรทเตอร์ ตัวโปรแกรมที่เป็นอินเทอร์เพรทเตอร์คือ Interpret ซึ่งเรื่องราวของโปรแกรมนี้ ได้กล่าวในเรื่องภาษาเครื่องสำหรับภาษาสูงในไมโครฯ 32 ซึ่งผมได้นำมาลงซ้ำอีกในรูป P โดยได้เพิ่มคำสั่ง

JTSR	**addr**	กระโดดไปยังโปรแกรมย่อย ณ แอดเดรส addr
RETS	**dist**	ส่งกลับจากโปรแกรมย่อยโดยยกเลิกตัวแปรในสแต็คเป็นจำนวน dist
MVST	**dist**	ย้ายข้อมูลตัวที่ dist จากยอดของสแต็คไปไว้ที่ชั้นยอดของสแต็ค

P

```
PROCEDURE  Interpret(prg : PRGtype);
CONST  MEMMAX = 1000;
VAR    PC, SP, adr : Integer;  ch : Char;
       endPRG, branch : Boolean;
{       mem : ARRAY[0..MAXMEM] OF Integer;  }


BEGIN  { Interpret }
  PC := 0;  SP := 499;  endPRG := FALSE;
  REPEAT
    branch := FALSE;
    IF  (PC > Ppntr) OR (PC < 0)  THEN
      BEGIN  WriteLn('PC error exit'); Exit  END;
    WITH  prg[PC]  DO
      BEGIN
      CASE  ncode  OF

ADDI : BEGIN  mem[SP-1] := mem[SP-1] + mem[SP]; SP := SP - 1  END;
SUBI : BEGIN  mem[SP-1] := mem[SP-1] - mem[SP]; SP := SP - 1  END;
MULI : BEGIN  mem[SP-1] := mem[SP-1] * mem[SP]; SP := SP - 1  END;
DIVI : BEGIN  mem[SP-1] := mem[SP-1] DIV mem[SP]; SP := SP - 1  END;
BEQL : IF  mem[SP-1] = mem[SP]  THEN  branch := TRUE;
BNEQ : IF  mem[SP-1] <> mem[SP]  THEN  branch := TRUE;
BGTR : IF  mem[SP-1] > mem[SP]  THEN  branch := TRUE;
BLSS : IF  mem[SP-1] < mem[SP]  THEN  branch := TRUE;
BGEQ : IF  mem[SP-1] >= mem[SP]  THEN  branch := TRUE;
BLEQ : IF  mem[SP-1] <= mem[SP]  THEN  branch := TRUE;
JUMP : branch := TRUE;
PUSH : BEGIN  SP := SP + 1; mem[SP] := oper  END;
PLUS : SP := SP + oper;
MINS : SP := SP - oper;
INCR : mem[SP] := mem[SP] + 1;
DECR : mem[SP] := mem[SP] - 1;
INNI : BEGIN SP := SP + 1; Read(mem[SP])  END;
INNS : BEGIN SP := SP + 1; Read(KBD,ch); Write(ch); mem[SP] := Ord(ch) END;
OUTI : BEGIN Write(mem[SP]); SP := SP - 1  END;
OUTS : BEGIN Write(Chr(mem[SP])); SP := SP - 1 END;
LODE : BEGIN adr := mem[SP]; SP := SP + 1; mem[SP] := mem[adr]  END;
STOR : BEGIN adr := mem[SP-1]; mem[adr] := mem[SP]; SP := SP - 1  END;
MOVE : BEGIN adr := mem[SP]; mem[SP] := mem[adr]  END;
HALT : endPRG := TRUE;
JTSR : BEGIN  SP := SP + 1; mem[SP] := PC; branch := TRUE  END;
RETS : BEGIN  PC := mem[SP]; SP := SP - (oper + 1)  END;
MVST : BEGIN  SP := SP + 1; mem[SP] := mem[SP - (oper)] END;
       END;  { CASE }
       IF  branch  THEN  PC := oper  ELSE  PC := PC + 1;
       END;  { WITH }
{DISPMEM;}
  UNTIL  endPrg;
WriteLn(' PC=',PC,'  SP=',SP);
END;
```

```
PROCEDURE ListCode(prg : PRGtype);
VAR  i : Integer;
BEGIN
  WriteLn; WriteLn('DISASMBLER...');
  FOR  i := 0 TO Ppntr DO
    WITH  prg[i]  DO
      IF  ncode IN needOperand  THEN
        WriteLn(i:3, InstrSet[ncode]:10, oper:5)
      ELSE
        WriteLn(i:3, InstrSet[ncode]:10);
END;
```

```
Program test one_one:  test assignment statement

PROGRAM  TestOneOne;
INT  num1, num2;
MAIN
  num1 = 100; WriteInt(num1); nl;
  num2 = 50 + 150; WriteInt(num2); nl;
  num1 = num2 + 100; WriteInt(num1); nl;
  ReadInt(num2); nl;
  num1 = num1 + num2; WriteInt(num1); nl;
ENDFTN
ENDPRO
```

รูปที่ 2 โปรแกรมทดสอบ การแปลประโยคให้ค่าแก่ตัวแปร

```
Test program one_tw0:  test if statement
PROGRAM  Test12
INT  alpha, bravo;

MAIN
  alpha = 50; WriteInt(alpha); nl;
  ReadInt(bravo); nl;
  IF  alpha > bravo  THEN
    alpha = alpha - bravo;
    WriteInt(alpha); nl;
    IF alpha < bravo  THEN
      alpha = bravo - alpha;
      WriteInt(alpha); nl;
    ENDIF
    WriteInt(bravo);
  ENDIF
ENDFTN
ENDPRO


B>
```

รูปที่ 3 โปรแกรมทดสอบ ประโยคโครงสร้าง IF ในแบบ IF ซ้อน

```
Enter source file name.IQ0  test11.iq0
PROGRAM TEST ONE_ONE:  TEST ASSIGNMENT STATEMENT

 PROGRAM  TESTONEONE;
 INT  NUM1, NUM2;
 MAIN
   NUM1 = 100; WRITEINT(NUM1); NL;
   NUM2 = 50 + 150; WRITEINT(NUM2); NL;
   NUM1 = NUM2 + 100; WRITEINT(NUM1); NL;
   READINT(NUM2); NL;
   NUM1 = NUM1 + NUM2; WRITEINT(NUM1); NL;
 ENDFTN
 ENDPRO
 Pp=61 Vcntr=3 Fcntr=0 memI=3
startPC = 0
Program are running..
100
200
300
50

350
  ...end of Program

>
```

รูปที่ 4 ผลการรันโปรแกรม TestOneOne เลข 50 คือ เลขที่ป้อนในขณะรัน

นอกจากนี้ยังได้เพิ่มการควบคุมการทำงานของโปรแกรมภาษา IQ0 คือ ถ้าค่า PC น้อยกว่า 0 หรือมากกว่า Ppntr คือ มากกว่าจำนวนที่ป้อนให้ยุติการรัน ท่านที่ใช้โปรแกรมเก่ากรุณาตรวจสอบด้วยครับ

สำหรับโปรแกรม ListCode ในรูป O ความจริงไม่ได้ใช้ในการแปลภาษา IQ0 ดังกล่าวมา ได้นำมารวมไว้ เพื่อบางท่านอาจต้องการดูว่า เมื่อคอมพายล์แล้วจะได้รหัสกลางเป็นอย่างไร และใช้ในการดีบัก (debug)

โปรแกรมทดสอบ

เป็นอันว่าเราได้สร้างภาษา IQ0 ขั้นพื้นฐานโดยใช้ชื่อโปรแกรมว่า IQ0-one เสร็จเรียบร้อยแล้วครับ ก็ควรจะได้ทดลองกันว่า โปรแกรมตัวแปรภาษานี้แปรถูกต้องหรือไม่ ได้ทำโปรแกรมทดสอบคือ โปรแกรม TestOneOne กับ Test12 ไว้ในรูปที่ 2 และที่ 3 และได้แสดงผลการรันโปรแกรม TestOne-One ไว้ในรูปที่ 4

ขอให้ป้อนโปรแกรม IQ0one และเขียนโปรแกรมทดสอบแบบต่างๆ รวมทั้งแบบที่ผิดไวยากรณ์ภาษา IQ0 แล้วทดสอบดู จะแสดงตำแหน่งที่ผิดได้ดีพอสมควรครับ

การดำเนินการต่อไป

ในฉบับต่อไปจะกล่าวถึงโครงสร้าง IF-ELSE และ WHILE รวมทั้งการดำเนินการกับ FUNCTION ขอให้ลองคิดดูครับว่าจะเขียนโปรแกรมได้อย่างไร สำหรับซอฟต์แวร์ประกอบบทความตอนนี้คงมีเท่าที่ผมกล่าวมานี้เท่านั้นยังไม่สมบูรณ์ครับ

หมายเหตุ : สำหรับผู้ที่ต้องการใช้โปรแกรม IQ0 ให้คีย์เรียงลำดับเริ่มจากรูป A ถึง Q